REMARKS

Claims 1-6 and 8-22 will be pending in the current Application upon entering this Amendment. Claims 1, 2, 4, 8, and 9 have been amended; claim 7 has been cancelled; and claims 17-22 have been added. Applicant submits that the amendments do not add new matter to the current Application. All the amendments herein have been made in order to clarify the claims and not for prior art reasons. Applicant also submits that (1) no amendment made was related to the statutory requirements of patentability unless expressly stated herein, and (2) no amendment made was for the purpose of narrowing the scope of any claim, unless Applicant has argued herein that such amendment was made to distinguish over a particular reference or combination of references.

Rejection of claims 1-4, 6, and 9-16 under 35 U.S.C. 102

Applicant respectfully submits that claims 1-4, 6, and 9-16 are patentable over US Patent No. 5, 701, 495 (hereinafter referred to as Arndt). Referring to claims 1 and 9, Applicant submits that claims 1 and 9, as amended, are patentable over Arndt. Applicant has amended claim 1 to include the limitations of dependent claim 7, and claim 9 has been amended to include limitations similar to that of claim 7. Therefore, amended claims 1 and 9 will be discussed in more detail below in response to the Examiner's 35 U.S.C. 103 rejection of claim 7.

Applicant submits that claim 14 is not anticipated by Arndt because Arndt does not teach or suggest each and every limitation of claim 14. For example, claim 14 requires executing software "to generate a predetermined software-generated interrupt signal which emulates a predetermined one of the hardware-generated interrupt sources but with a priority which differs from the predetermined one of the hardware-generated interrupt sources." However, Arndt makes no mention about using software-generated interrupt signals to emulate hardware-generated signals. Arndt only provides a method of routing interrupt signals in order to present them to the proper processor. The methods of Arndt do not provide a method for dynamically changing prioritization of servicing of interrupts. The Examiner cites col. 9, lines 15-67, col. 10, lines 1-76, and col. 11, lines 1-14 of Arndt as teaching this limitation; however, these portions discuss the overall flow of interrupt handling. Although priorities of the interrupts are used to properly route and present both hardware and software interrupts in Arndt, Arndt does not teach

or suggest generating a predetermined software-generated interrupt signal which emulates a predetermined one of the hardware-generated interrupt sources. Therefore, Applicant submits that claim 14 is allowable over Arndt. Claims 15-16 depend from allowable claim 14 and are therefore allowable for at least those reasons stated above with respect to claim 14.

Note that added claim 22 is a combination of previous claim 1 and claim 6. Claim 22 requires changing prioritization level of a predetermined hardware-generated interrupt by providing a software-generated interrupt which represents a corresponding hardware-generated interrupt sources for the predetermined hardware-generated interrupt. Arndt also does not teach or suggest changing prioritization levels as claimed in claim 22; therefore, Applicant submits that new claim 22 is also allowable over Arndt.

Note also that claim 4 has been amended to correct a typographical error where "same" has been changed to "some."

Rejection of claims 5, 7, and 8 under 35 U.S.C. 103

Applicant respectfully submits that claims 5, 7, and 8 are patentable over Arndt in view of US Patent No. 6, 412, 081 (hereinafter referred to as Koscal). As stated above, Applicant has amended claim 1 to include the limitations of dependent claim 7, and claim 9 has been amended to include limitations similar to that of claim 7. Furthermore, Applicant submits that claims 1 and 9, as amended, are patentable over Arndt in view of Koscal because neither Arndt, Koscal, nor their combination teach or suggest every limitation of claims 1 or 9.

Claim 1 now requires determining priority between two interrupts, a first interrupt being hardware-generated and a second interrupt being software-generated, when the two interrupts have a same prioritization level (as previously required in dependent claim 7). The Examiner sates that the predetermined bit of the processor status register (PSR) of Koscal teaches this limitation. However, Applicant respectfully disagrees. As used in Koscal, the PSR is a register in which the status of the microprocessor is maintained. The predetermined bit of the PSR referred to by the Examiner is only used to *distinguish* between 1) a software interrupt or an interrupt generated responsive to a trap condition, and 2) a hardware interrupt. That is, the predetermined bit only distinguishes the current interrupt and does not determine priority. Also, the Examiner cites the compare circuitry of col. 7, lines 49-63, of Koscal. However, the compare circuitry simply compares a trap address to the address data placed on the address bus to

selectively assert a trap condition signal if a trap has occurred. Therefore, the compare circuitry of Koscal also does not teach or suggest determining priority between two interrupts as claimed in claim 1. Claims 2-6 and 17 depend directly or indirectly from allowable claim 1 and are therefore allowable for at least those reasons stated with respect to claim 1.

Claim 9 has been amended to further define the logic circuitry as determining priority between two interrupts, a first interrupt being hardware-generated and a second interrupt being software-generated, when the two interrupts have a same prioritization level. Therefore, for at least those reasons mentioned in reference to claim 1, claim 9 is also allowable over Arndt in view of Koscal.

Claim 8 has also been rewritten in independent form, and Applicant submits that claim 8 is allowable over Arndt in view of Koscal, because neither Arndt, Koscal, nor their combination teaches or suggests coupling enabling circuitry between the first and second storage devices and the logic circuitry, as claimed in claim 1. As discussed above, the predetermined bit in the PSR only distinguishes between different types of interrupts (software interrupt or one responsive to a trap condition versus a hardware interrupt). Also, the compare circuitry of Koscal does not teach or suggest the enabling circuitry of claim 8 which is used to determine whether to pass the hardware-generated and software-generated interrupts to the logic circuitry for further processing. The compare circuit of Koscal is used to detect the occurrence of a trap condition and asserts the trap condition signal if one occurs. This information is then used to distinguish a trap condition from a software interrupt and is not used to determine whether to pass the hardware-generated and software-generated interrupts to the logic circuitry for further processing. Therefore, the compare circuit and the PSR are used to *distinguish* between types of interrupts to determine appropriate interrupt routines. However, they do not teach or suggest the limitations of claim 8. Therefore, Applicant submits that claim 8 is allowable over Arndt in view of Koscal. Claims 18-21 depend directly or indirectly from allowable claim 8 and are therefore allowable for at least those reasons mentioned above with respect to claim 8.

Conclusion

Although Applicant may disagree with statements made by the Examiner in reference to the claims and the cited references, Applicant is not discussing all these statements in the current Office Action, yet reserves the right to address them at a later time if necessary.

Applicant respectfully solicits allowance of the pending claims. Contact me if there are any issues regarding this communication or the current Application.

If Applicant has overlooked any additional fees, or if any overpayment has been made, the Commissioner is hereby authorized to credit or debit Deposit Account 502117.

Respectfully submitted,

By: _____

Joanna G. Chiu
Attorney of Record
Reg. No.: 43,629
Telephone: (512) 996-6839
Fax No.: (512) 996-6854

SEND CORRESPONDENCE TO:

Motorola, Inc.
Law Department

Customer Number: 23125

CLAIMS - VERSION WITH MARKINGS TO SHOW CHANGES MADE

(note -- a full set of claims is being provided for the convenience of the Examiner *where claims 1, 2, 4, 8, and 9 have been amended herein and claims 17-22 have been added herein.*)

1. **(Amended)** A method for implementing interrupts in a data processing system, comprising the steps of:

providing a first storage device having a plurality of inputs, each of the plurality of inputs being coupled by a respective physical conductor to one of a plurality of hardware-generated interrupt sources which selectively generate hardware interrupts and selectively storing the hardware interrupts, the first storage device providing one or more hardware-generated interrupt signals;

providing a second storage device having one or more inputs, each of the one or more inputs receiving and storing a predetermined one of a plurality of software-generated interrupt signals, at least some of the predetermined plurality of software-generated interrupt signals indicating an interrupt from a different source or of a different type than the hardware interrupts, the second storage device providing one or more software-generated interrupt signals; [and]

coupling logic circuitry to the first storage device and the second storage device for receiving the one or more hardware-generated interrupt signals and the one or more software-generated signals, the logic circuitry providing an interrupt request signal which will cause an interrupt to occur in the data processing system; and

determining priority between two interrupts, a first interrupt being hardware-generated and a second interrupt being software-generated, when the two interrupts have a same prioritization level by choosing to service one of the hardware-generated first interrupt or the software-generated second interrupt.

2. (**Amended**) The method of claim 1 further comprising the step of:

assigning an interrupt prioritization level to [specific] storage locations of the first storage device and the second storage device[, the interrupt prioritization level of the plurality of hardware-generated interrupt sources coupled to the first storage device being permanently assigned, but assignment of the interrupt prioritization level of interrupt sources associated with the second storage device being variable by software control].

3. The method of claim 2 further comprising the step of:

assigning a portion of the plurality of software-generated interrupt signals stored in the second storage device to represent interrupts from some interrupt sources generating hardware interrupts and having a corresponding interrupt prioritization level.

4. (**Amended**) The method of claim 2 further comprising the step of:

assigning a portion of the plurality of software-generated interrupt signals stored in the second storage device to represent interrupts from [same] <u>some</u> interrupt sources generating hardware interrupts and having an interrupt prioritization level which differs from the interrupt prioritization level of the plurality of hardware-generated interrupt sources coupled to the first storage device.

5. The method of claim 2 further comprising the step of:

changing interrupt servicing from servicing a hardware-generated interrupt and switching to servicing a software-generated interrupt of higher prioritization before completion of servicing of the hardware-generated interrupt occurs.

6. The method of claim 2 further comprising the step of:

changing prioritization level of a predetermined hardware-generated interrupt by providing a software-generated interrupt which represents a corresponding

hardware-generated interrupt source for the predetermined hardware-generated interrupt but with a different prioritization level than the predetermined hardware-generated interrupt.

8. (**Amended**) A method for implementing interrupts in a data processing system, comprising the steps of:

providing a first storage device having a plurality of inputs, each of the plurality of inputs being coupled by a respective physical conductor to one of a plurality of hardware-generated interrupt sources which selectively generate hardware interrupts and selectively storing the hardware interrupts, the first storage device providing one or more hardware-generated interrupt signals;

providing a second storage device having one or more inputs, each of the one or more inputs receiving and storing a predetermined one of a plurality of software-generated interrupt signals, at least some of the predetermined plurality of software-generated interrupt signals indicating an interrupt from a different source or of a different type than the hardware interrupts, the second storage device providing one or more software-generated interrupt signals;

coupling logic circuitry to the first storage device and the second storage device for receiving the one or more hardware-generated interrupt signals and the one or more software-generated signals, the logic circuitry providing an interrupt request signal which will cause an interrupt to occur in the data processing system; and

[The method of claim 1 further comprising the step of:]

coupling enabling circuitry between the first and second storage devices and the logic circuitry, the enabling circuitry receiving the hardware-generated and software-generated interrupts and determining whether to pass the hardware-generated and software-generated interrupts to the logic circuitry for further processing.

9. (**Amended**) A data processing system with interrupt control circuitry, comprising:

a plurality of hardware interrupt sources;

a hardware interrupt storage device having a plurality of inputs, each of the plurality of inputs being coupled by an electrical conductor to one of a plurality of hardware interrupt sources, the hardware interrupt storage device storing hardware-generated interrupts and providing each of the hardware-generated interrupts at a predetermined output terminal;

a software interrupt storage device having a plurality of inputs, each of the plurality of inputs receiving a predetermined one of a plurality of software-generated interrupt signals, at least one of the software-generated interrupt signals corresponding to interrupt servicing of a portion of the data processing system which is not designated as a hardware interrupt source; and

logic circuitry coupled to the hardware interrupt storage device and the software interrupt storage device for providing a data processing system interrupt signal in response to receipt of either hardware-generated interrupts or software-generated interrupts, wherein the logic circuitry determines priority between two interrupts, a first interrupt being hardware-generated and a second interrupt being software-generated, when the two interrupts have a same prioritization level by choosing to service one of the hardware-generated first interrupt or the software-generated second interrupt.

10. The data processing system of claim 9 wherein the hardware interrupt storage device and the software interrupt storage device have an assigned interrupt prioritization level to specific storage locations, the interrupt prioritization level of the hardware interrupt sources being permanently assigned, but assignment of the interrupt prioritization level of interrupt sources associated with the software-generated interrupt signals being variable by software control.

11. The data processing system of claim 10 wherein a software-generated interrupt signal of higher priority than a currently executing hardware-generated interrupt signal is provided to the logic circuitry prior to completion of an associated hardware interrupt servicing, and the data processing system suspends processing of the hardware interrupt servicing to process an associated software interrupt servicing.

12. The data processing system of claim 9 further comprising:

a mask register coupled to the hardware interrupt storage device and the software interrupt storage device for selectively preventing hardware-generated interrupt signals and software-generated interrupt signals from propagating to the logic circuitry.

13. The data processing system of claim 9 wherein the hardware interrupt storage device and the software interrupt storage device are each implemented as latch circuits.

14. A method for implementing interrupts in a data processing system, comprising the steps of:

providing a first storage device having a first plurality of prioritized storage locations representative of priority of interrupt servicing for hardware-interrupt signals stored therein, the first storage device having a plurality of inputs, each of the plurality of inputs being coupled by a physical conductor to a plurality of hardware-generated interrupt sources which selectively generate hardware interrupts and storing the hardware interrupts, the first storage device providing one or more hardware-generated interrupt signals;

providing a second storage device having a second plurality of prioritized storage locations representative of priority of interrupt servicing for software-generated interrupt signals stored therein, the second storage device having one or more inputs, each of the one or more inputs receiving and storing a predetermined one of a plurality of software-generated interrupt signals, the predetermined plurality of software-generated interrupt signals

indicating an interrupt from a different source or of a different type than the hardware interrupts, the second storage device providing one or more software-generated interrupt signals;

executing software with the data processing system to generate a predetermined software-generated interrupt signal which emulates a predetermined one of the hardware-generated interrupt sources but with a priority which differs from the predetermined one of the hardware-generated interrupt sources, thereby dynamically changing prioritization of servicing of interrupts in the data processing system; and

coupling logic circuitry to the first storage device and the second storage device for receiving the one or more hardware-generated interrupt signals and the one or more software-generated signals, the logic circuitry providing an interrupt request signal which will cause an interrupt to occur in the data processing system.

15. The method of claim 14 further comprising the steps of:

generating the predetermined software-generated interrupt signal which emulates the predetermined one of the hardware-generated interrupt sources while another hardware-generated interrupt is being serviced, the predetermined software-generated interrupt signal having a priority which is higher than the other hardware-generated interrupt being serviced; and

suspending servicing of the other hardware-generated interrupt being serviced to begin servicing of the predetermined software-generated interrupt signal.

16. The method of claim 14 further comprising the step of:

masking the one or more hardware-generated interrupt signals and the one or more software-generated interrupt signals to selectively pass active interrupt signals to the logic circuitry in response to an enable signal.

17. **(New Claim)** The method of claim 2 further wherein the interrupt prioritization level is assigned to specific storage locations of the first and second storage device, the interrupt prioritization level of the plurality of hardware-generated interrupt sources coupled to the first storage device being permanently assigned, but assignment of the interrupt prioritization level of interrupt sources associated with the second storage device being variable by software control.

18. **(New Claim)** The method of claim 8 further comprising the step of:
assigning an interrupt prioritization level to storage locations of the first storage device and the second storage device.

19. **(New Claim)** The method of claim 18 further comprising the step of:
assigning a portion of the plurality of software-generated interrupt signals stored in the second storage device to represent interrupts from some interrupt sources generating hardware interrupts and having a corresponding interrupt prioritization level.

20. **(New Claim)** The method of claim 18 further comprising the step of:
assigning a portion of the plurality of software-generated interrupt signals stored in the second storage device to represent interrupts from some interrupt sources generating hardware interrupts and having an interrupt prioritization level which differs from the interrupt prioritization level of the plurality of hardware-generated interrupt sources coupled to the first storage device.

21. **(New Claim)** The method of claim 18 further comprising the step of:
changing prioritization level of a predetermined hardware-generated interrupt by providing a software-generated interrupt which represents a corresponding hardware-generated interrupt source for the predetermined hardware-generated interrupt but with a different prioritization level than the predetermined hardware-generated interrupt.

22. **(New Claim)** A method for implementing interrupts in a data processing system, comprising the steps of:

providing a first storage device having a plurality of inputs, each of the plurality of inputs being coupled by a respective physical conductor to one of a plurality of hardware-generated interrupt sources which selectively generate hardware interrupts and selectively storing the hardware interrupts, the first storage device providing one or more hardware-generated interrupt signals;

providing a second storage device having one or more inputs, each of the one or more inputs receiving and storing a predetermined one of a plurality of software-generated interrupt signals, at least some of the predetermined plurality of software-generated interrupt signals indicating an interrupt from a different source or of a different type than the hardware interrupts, the second storage device providing one or more software-generated interrupt signals;

coupling logic circuitry to the first storage device and the second storage device for receiving the one or more hardware-generated interrupt signals and the one or more software-generated signals, the logic circuitry providing an interrupt request signal which will cause an interrupt to occur in the data processing system; and

changing prioritization level of a predetermined hardware-generated interrupt by providing a software-generated interrupt which represents a corresponding hardware-generated interrupt source for the predetermined hardware-generated interrupt but with a different prioritization level than the predetermined hardware-generated interrupt.